Wireless Networks (2020) 26:4319–4338
https://doi.org/10.1007/s11276-020-02331-1

RLProph: a dynamic programming based reinforcement learning approach for optimal routing in opportunistic IoT networks

Deepak Kumar Sharma[1] · Joel J. P. C. Rodrigues[2,3] · Vidushi Vashishth[1] · Anirudh Khanna[4] · Anshuman Chhabra[4,5]

Published online: 28 April 2020

Abstract

Routing in Opportunistic Internet of Things networks (OppIoTs) is a challenging task because of intermittent connectivity between devices and the lack of a fixed path between the source and destination of messages. Recently, machine learning (ML) and reinforcement learning (RL) have been used with great success to automate processes in a number of different problem domains. In this paper, we seek to fully automate the OppIoT routing process by using the Policy Iteration algorithm to maximize the possibility of message delivery. Moreover, we model the OppIoT environment as a Markov decision process (MDP) replete with states, actions, rewards, and transition probabilities. The proposed routing protocol, RLProph, is able to optimize the routing process via the optimal policy obtained by solving the MDP using Policy Iteration. Through extensive simulations, we show that RLProph outperforms a number of ML-based and context-aware routing protocols on a multitude of performance criteria.

Keywords Opportunistic networks · Internet of Things · Reinforcement learning · Markov decision process · Dynamic programming · ONE simulator · Machine learning · Policy iteration

1 Introduction

In traditional Mobile Ad-hoc Networks (MANETs), transmitting a message from a sender to a receiver device is usually undertaken only if an end-to-end connection between the transmitting and receiving pair exists. However, in situations where such an end-to-end connection is not realistically achievable, such as in disaster management scenarios, rural areas, and wildlife monitoring networks, the usage of MANETs is not a viable option. Opportunistic Networks (OppNets) are a subclass of Delay Tolerant Networks (DTNs) which meet this requirement for such specific networking applications. Thus, they can be used in situations where communication opportunities are intermittent, and an end-to-end path between the source and the destination may never exist. Due to these advantages, OppNets have seen many real-world applications recently, such as ZebraNet [1], DakNet [2], and the Sami Network Connectivity (SNC) project [3], among others. This unique characteristic of OppNets also extends to the domain of Internet of Things networks. The Opportunistic Internet of Things (OppIoT) networks [4] were introduced

✉ Deepak Kumar Sharma
dk.sharma1982@yahoo.com

Joel J. P. C. Rodrigues
joeljr@ieee.org

Vidushi Vashishth
vidushivashishth96@gmail.com

Anirudh Khanna
anirudhkhanna9654@gmail.com

Anshuman Chhabra
chhabra@ucdavis.edu

1 Department of Information Technology, Netaji Subhas University of Technology, New Delhi, India

2 Federal University of Piauí (UFPI), Campus Petrônio Portela, Ininga, Teresina, PI, Brazil

3 Instituto de Telecomunicações, Covilhã, Portugal

4 Division of Electronics and Communication Engineering, Netaji Subhas University of Technology, New Delhi, India

5 Department of Computer Science, University of California, Davis, USA

to explore information dissemination in IoT through variable contact opportunities between humans. OppIoT helps overcome geographical limitations pertaining to network architecture availability. Owing to the similarity in message routing behavior, algorithms applicable to OppNets also work for OppIoT.

In an OppIoT network, routing is performed using the *store-carry-forward* principle where the source node waits for opportunities to forward the message to *relay nodes*, which store and carry it. Relay nodes in turn seek to further forward the message to other relays in their proximity, leading to eventual delivery of the message to its destination. Thus, as networking in an OppIoT network is inherently intermittent, routing messages from source to destination can become a daunting and challenging task. Moreover, it is important to devise routing protocols that are capable of increasing the possibility of eventual message delivery and reducing chances of delivery failure.

Routing protocols for OppNets and OppIoT networks can be classified into two types: *context-oblivious* and *context-aware*. In context-oblivious routing, context information about the nodes and the network is ignored while making routing decisions. Thus, in these algorithms nodes generally aim to deliver messages without using network-specific information that can improve message delivery. Therefore, message delivery probability for such protocols is generally low. Moreover, for flooding-based routing protocols that come under this category (for example, Epidemic Routing [5]) network overhead and resource consumption are generally very high. On the other hand, context-aware protocols use context information to identify the best possible relays and make informed routing decisions. Since the decisions consider node specific information that can help with routing, these routing protocols generally tend to perform better than context-oblivious protocols. Some examples include PRoPHET [6] and HBPR [7, 8].

Some routing protocols also incorporate the use of Machine Learning (ML) in making routing decisions. For example, MLProph [9] uses Neural Network and Decision Tree based ML techniques to first train on past network routing data, and then make efficient routing decisions by predicting the delivery probability. GMMR in [10] uses Gaussian Mixture Models to perform soft clustering based routing in OppIoT networks. Recently, Reinforcement Learning (RL), a sub-class of ML, has been very successfully used in many autonomous applications: achieving near-perfect scores in Atari video games [11] and mastering the game of Go (AlphaGo) [12]. RL is concerned with how *agents* or *players* in a system should take decisions or *actions* to achieve certain desired behavior, which is generally indicated by associated *rewards*. RL trains the agent to exhibit desired behavior by maximizing the rewards associated with the actions taken by the agent. In this paper, we use a type of RL approach, called *planning*, to automate the routing decision problem, and ensure a high-rate of message delivery. We describe our approach in detail in Sect. 3.

Most of the routing protocols that we have discussed so far make some assumptions about the network environment and then make routing decisions based on these assumptions. For example, PRoPHET assumes that message delivery is correlated to node history, and a transitivity property among nodes, and then predicts whether or not the message will be delivered based on these factors. While these presumptions about the network environment might hold true, there might be unforseen factors that can help make improved routing decisions which might be ignored by these protocols. Moreover, the method by which routing decisions are generally made in some protocols is often constrained to some threshold or a human-chosen factor, and only if these conditions are met, is routing performed. For example in MLProph, only if the predicted delivery probability is greater than a threshold of 0.8, is the message routed to the potential relay. Thus, we propose to ameliorate these problems by using a RL planning approach which learns to make the best possible routing decisions for nodes (agents) in the environment over time, by itself. Therefore, instead of making unjustifiable assumptions about the way routing should be performed in an OppIoT network, we design a routing protocol that considers the OppIoT network as a system that learns to optimally route messages amongst nodes by itself. Since RL problems aim to optimize certain requirements, in our approach we seek to maximize message delivery and minimize the overhead on the network.

For any RL problem, a Markov Decision Process (MDP) [13] needs to be formulated for the system first. A MDP is thus a mathematical framework for formulating discrete time stochastic control and RL problems. A MDP is defined as a tuple of five values (S, A, P_a, R_a, γ): S is a finite set of states describing the environment, A is the finite set of actions that agents in the environment can take, $P_a(s, s') = \mathbb{P}(s_{t+1} = s' | s_t = s, a_t = a)$ is the probability that action a ($a \in A$) in state s ($s \in S$) at time t will lead to state s' at time $t+1$, R_a is the immediate reward received after transitioning from state s to state s' due to action a, and γ is the discount factor representing the difference in value between future rewards and present rewards.

In our approach, designing the MDP entails modeling the OppIoT environment as a set of all possible states that nodes in the network can be in. These states are described by using certain features or node specific contextual information. The actions that make up the MDP are also directly related to the routing problem: either choosing to relay a message to a neighboring node or choosing not to.

The rest of the MDP is structured to improve message delivery using both the reward function and state transition probabilities. This is achieved by assigning larger rewards to states with higher possibility of message delivery, and by designating higher transition probabilities from states of low message delivery probability to states of high message delivery probability. Once the MDP is designed, a Dynamic Programming based algorithm, called Policy Iteration [14] is used for finding the optimal policy that describes how routing should be performed in the network. Our routing protocol is explicitly detailed in Sect. 3. Furthermore, to the best of our knowledge, this is the first time an RL planning based approach has been used for routing in OppIoT. The major contributions of this paper can be summarized as follows:

- Modeling the OppIoT environment as a MDP to optimally route messages from source to destination
- Defining a suitable reward function in the MDP that is capable of improving message delivery probability and reducing network overhead
- Solving the MDP using Policy Iteration to get the optimal policy which is used to design the routing protocol and obtain state-of-the-art results

The rest of the paper is organized into different sections as follows: Sect. 2 elaborates on the background and related work on routing protocols in OppIoT. Section 3 outlines the proposed approach of the RL based routing protocol. Section 4 covers the results and comparisons obtained after conducting extensive simulations of the proposed protocol and a few existing protocols on the ONE simulator. Finally, Sect. 5 concludes the paper and highlights the scope for future work.

2 Background and related work

2.1 Context-aware routing protocols

Context-aware routing protocols utilize certain contextual information about the nodes of the network for making their routing decisions. These algorithm use information or features from the nodes as input for selecting the next hop for a message. Protocols such as PRoPHET [6], PRoPHET+ [15], PROPICMAN [16], MobySpace [17], MV [18], HiBOp [19], PRoWait [20], HBPR [7, 8], CAOR [21], KNNR [22], MLProph [9], CPTR [23], DAS [24], Social Similarities based adaptive routing in [25], forwarder selection algorithm in [26], and data transmission algorithm in [27] are a few examples of context-aware protocols.

PRoPHET uses the history of encounters among nodes and a transitive property mathematical formulation to compute a probabilistic estimate of message delivery. This information is used while choosing a suitable node for forwarding the message. PRoPHET+ improves upon PRoPHET and utilizes the nodes' buffer size, power, location, popularity and the predictability value as a weighted function to make better predictions. In PROPICMAN the sender node distributes each message's header to its neighbors. The header contains information that the sender knows about the destination. This information is used as a basis for computation of delivery probability by the neighbors. The sender then transmits the message on the two-hop route(s) with the highest delivery probability.

MobySpace and MV are routing protocols which take advantage of nodes' mobility while making routing decisions. The authors of MobySpace suggest that two nodes having similar mobility patterns are more likely to meet each other, thus enhancing communication. Based on this principle a Euclidean virtual space, also known as MobySpace, is formed. Nodes make their decisions based on the MobySpace. Messages are forwarded to those nodes that have the most common mobility patterns with respect to the destination node. The routing protocol MV utilizes the frequency of nodes meeting each other and their visits to a particular region in the geographical grid. When a source node comes in contact with a relay node in MV, they exchange some information. This information contains the list of messages that node carries as well as it's subsequent destination. Both nodes calculate the likelihood of forwarding each other's messages based on a derived formula. After this step, nodes sort their lists by the likelihood of delivery and also delete the messages that another node has a higher probability of delivering.

HiBOp and HBPR extract information from the history of nodes' visits. HiBOp stores information in two tables, namely, Identity Table (IT) and History Table (HT). IT describes the current context of the node which includes information about the current neighbors of the node. Since the information conveyed by the current context is limited, it becomes essential to store the history of node encounters as well. Thus, HT helps store past node information originally populating the IT. HiBOp makes its decisions by exploiting this information to choose the best forwarder. HBPR divides the entire network area into cell sections. Each node uses two tables to make routing decisions. A history table is used to store the record of the node's location and timestamp. A home location table is also used to store home locations of other nodes in the network. Along with this data, nodes' speed and direction of movement are utilized to enhance routing decisions.

CAOR exploits various types of context information like nodes' energy, distance progress, node mobility, link quality etc. It also applies the Analytic Hierarchy Process theory to adjust the weights of context information to fit with the protocol behavior. PRoWait is a hybrid forwarding scheme that combines PRoPHET and Spray and Wait protocols. Selection of a suitable forwarder is achieved using PRoPHET, and the message packets are forwarded to the neighboring nodes using Spray and Wait.

KNNR and MLProph incorporate machine learning in their routing algorithms to make decisions. KNNR performs the routing operation in two phases: training phase and application phase. It first stores nodes' behavior in a dataset which includes features like neighbors' distance, hop count, interaction probability, among others. The protocol then trains a K-Nearest Neighbors classifier to be used in the application phase. During the application phase, the decision of routing the message to a particular relay is made using the trained classifier from the previous phase. MLProph uses two classification algorithms: Decision Trees and Neural Networks in its protocol. In its execution, MLProph trains itself based on various factors such as buffer capacity, hop count, node energy, popularity parameter, and predictability value, which are features inherited from the PROPHET routing scheme. As mentioned before, MLProph assumes a threshold delivery probability of 0.8 for any message to be forwarded to the next relay node. The proposed protocol in this paper does not make such assumptions.

2.2 Context-oblivious routing protocols

Context-oblivious protocols do not utilize context information and generally incorporate some aspect of message flooding as a means of routing. The Epidemic [5] routing protocol is the most widely employed context-oblivious protocol. It does not utilize network parameters or any other network related information to influence the routing decision. Spray and Wait [28] is another context-oblivious routing protocol. It executes controlled flooding by first *spraying* some copies in the network and then *waiting* till one of them reaches the destination. Spray and Focus [29] improves upon Spray and Wait by using a single copy based utility scheme which exploits the advantages of controlled replication. It fastens the delivery rate by finding better forwarding opportunities for nodes. These protocols generally perform poorly in terms of message delivery probability when compared to context-aware protocols.

3 Proposed approach

Prior to explaining the proposed approach, it is imperative to understand RL approaches to autonomous learning. A RL problem can be understood as the feedback system shown in Fig. 1. Here the agent chooses certain actions based on the states of the environment as well as the reward it obtains from taking certain actions in the previous time-step. Therefore, this continuous loop of choosing actions for states that maximize the total possible reward is how an agent *learns* to optimize certain desirable behavior. The environment and all related variables in a RL problem (rewards, states, and actions) are modeled as a discrete-time Markov Decision Process. Moreover, after constructing the MDP, RL problems can either be solved in an *online* or an *offline* fashion. Here, *online* refers to training on-the-go, and making decisions that affect the agent in real-time using algorithms such as Q-Learning and Policy Gradients. The other alternative, which is better suited to the OppIoT routing problem, is offline learning (or *planning*), where the optimal choice of actions for states (optimal policy) are first computed for the MDP using Dynamic Programming based algorithms such as Policy Iteration, Value Iteration, and others. Offline learning is a more desirable alternative for OppIoT routing as it has minimal overhead in terms of memory and computation, as once the optimal policy has been computed, agents are just supposed to follow it. On the other hand, online learning algorithms have large requirements for computation and processing, which OppIoT nodes might not possess owing to memory and computation power constraints in mobile devices.

As mentioned before, we propose using a RL planning approach to automate the routing process in our routing protocol. The basis of our RL problem first requires formulating a MDP that accurately captures the OppIoT environment and then solving the MDP to obtain an optimal policy that makes improved routing decisions. In this section, we explain the working of our proposed protocol which from hereon will be referred to as RLProph.

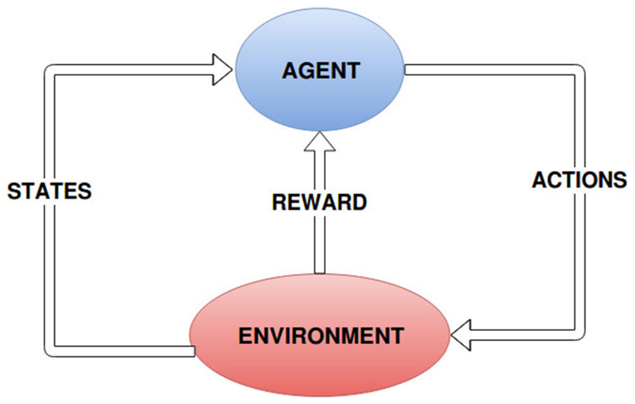


Fig. 1 Schematic explaining the working of a RL system

3.1 Constructing the MDP

To construct a MDP for the OppIoT environment, we first need to define states. The states essentially describe the current behavior or characteristics of devices in the OppIoT network. Moreover, in a MDP, at any time, transitions from a state (s) to a new state (s') are entirely possible, by performing a certain action (a). The system is given a reward (R_a) when the state changes depending on whether this is desirable behavior or not. Rewards can be lower and higher, or positive and negative to promote the desired behavior that needs to be learned by the system. Thus, the reward function for assigning R_a also needs to be formulated. Moreover, the probability of transition between states (P_t) is also a defining characteristic of MDPs, and is based on the action taken (a) as well as the current state. Thus the probability of transition from a state s to a new state s' is a function of s and the action taken, a.

In the subsections that follow, we define each element of the MDP five-tuple, and explain their working in improving routing in OppIoT networks.

3.1.1 States (S)

S is the set of all states that nodes in the MDP can be in, at any given time. States in a MDP provide information about the environment and help describe it in a discretized manner. When making a decision regarding the choice of action to take (in our case, choosing to route or not to route a message to a particular relay), the agent (routers) considers the state that the nodes are in before it does so. Therefore, states influence the decision making process undertaken by the agent in the RL system. The agent in this case are the nodes (routers) which follow the computed policy to make this decision.

Since states in a MDP essentially describe the RL system (in this case the OppIoT network) based on contextual information, we choose certain features that possibly capture all relevant information regarding the environment. Thus, we consider the following five features for modeling the states of an OppIoT environment:

- *PRoPHET probability* It signifies the probability of selecting next node as proposed in the well-performing PRoPHET routing protocol. High values of PRoPHET Probability indicate that there is a higher chance of message delivery by this particular node.
- *Buffer occupancy* It is a measure of the space left in the buffer to accommodate more packets of message data. Low Buffer Occupancy is desirable for better performance as it leads to lower network overhead. Moreover, a node with low buffer occupancy is less likely to drop a relayed message.
- *Successful deliveries* This feature describes the number of successful message deliveries made by this particular sender-receiver node pair. Higher number of prior successful deliveries generally lead to more deliveries in the future.
- *Distance to destination* It represents the physical distance between the relay node and the destination node. The closer the relay node is to the destination, the higher chance there is of actual message delivery.
- *Message live time* It is the time duration for which the message has been *live*, or the time passed since the message was first created by the source node. Lower values of Message Live Time ensure lower network overhead experienced.

We generate a dataset for these features by running a simulation on the ONE simulator [30]. We use Epidemic router to run this simulation. From the node encounters in this simulation, we extract continuous data values for each of the aforementioned features, giving us a data feature matrix of size 5280×5. However, since MDP can only understand discrete-time states, we need to convert the data matrix values into discrete values to obtain MDP states. To do this, we employ a process known as *discretization* or *binning*, which is used for converting continuous data into their discrete equivalents.

Discretization can be done on the basis of size, frequency or on the basis of some defined contextual specifications [31–33]. Since having a very large number of states in a MDP generally adds considerable memory overhead and increases the time for optimal policy computation, we utilize a simple equal-frequency binning approach for obtaining the states. Moreover, adding states can increase memory requirements exponentially, as the size of the matrix holding the transition probability values varies as the square of the number of states in the MDP. We decided to divide each of our five features into four individual *bins* or divisions, until and unless the value distribution for that feature could be accommodated in less than four bins. The reason for binning this way is that the maximum number of possible states would then be 1024 (4^5) which is a relatively workable number of states for our experiments. However, this value has been chosen this way just for our experiments, and can be specified differently by the algorithm designer to suit the needs of their network. Irrespective of our choices, we show through extensive simulations in the Results section that RLProph works robustly compared to other OppIoT and OppNet routing protocols.

Moreover, equal-frequency binning dictates that each bin of a feature should be sized equally and contain the same number of values as the other bins for that feature. For this, we sort each feature column independently and find four range divisions in increasing order, to bin the data into. The maximum-minimum values obtained for each of the features are shown in Table 1. The general algorithm for equal-frequency binning for N bins is shown in Algorithm 1. We also find that the feature Successful Deliveries, or feature 3, is better suited to discretization in 3 bins. Therefore, the features are discretized as follows:

1. Feature 1 (PRoPHET Probability) $\rightarrow$ 4 bins
2. Feature 2 (Buffer Occupancy) $\rightarrow$ 4 bins
3. Feature 3 (Successful Deliveries) $\rightarrow$ 3 bins
4. Feature 4 (Distance To Destination) $\rightarrow$ 4 bins
5. Feature 5 (Message Live Time) $\rightarrow$ 4 bins

The number of states that populate our MDP are calculated as: $4 * 4 * 3 * 4 * 4 = 768$. Thus, our constructed MDP possesses 768 states which model the OppIoT environment.

Table 1 Range of features used in dataset

Parameter	Minimum Value	Maximum Value
PRoPHET probability	0.0	1.0
Buffer occupancy (bytes)	636.0	49500000.0
Successful deliveries	0.0	2208.0
Distance to destination (m)	0.0	3684.84244
Message live time (s)	0.0	17936.8

Algorithm 1 Equal-Frequency Discretization Algorithm (For N Bins)

```
for each feature used do
    for each sample of specific feature do
        if sample value lies in range1 then
            Assign bin 1
        else if sample value lies in range2 then
            Assign bin 2
        else if sample value lies in range3 then
            Assign bin 3
    ...
    ...
    ...
        else if sample value lies in range$N$ then
            Assign bin $N$
```

3.1.2 Actions (*A*)

A is the set of all possible actions that can be performed by the device nodes in the OppIoT as part of the MDP structure. To accurately model the routing problem that we seek to optimize, we define the action space as a set of two actions: either choosing to forward the message to the potential relay node in proximity, or not choosing to do so. These binary actions are symbolized as 0 and 1, respectively. Therefore, the set of all possible actions, A, for a transmitting node in a node encounter with a potential relay, are as follows:

- $0 \rightarrow$ Forward the message to this relay node
- $1 \rightarrow$ Do not forward the message to this relay node and wait for another potential relay

3.1.3 Reward function (*R*)

The reward function in a MDP is formulated to promote and reward desirable behavior that the agent should learn autonomously, over time. The reward function assigns a reward when the agent performs an action to transition to a new state from it's current state. Thus, the reward function utilizes both the current state, as well as the action performed to assign an appropriate reward to the learning system. In our problem, we wish to positively reward behavior that improves message delivery probability and network overhead, and negate behavior that works contrarily to these objectives. Mathematically, the reward function can be represented as follows:

$$R(s,a) = \mathbb{E}[R_{t+1} | S_t = s, A_t = a] \tag{1}$$

where $R(s, a)$ represents the reward received after transitioning to another state given that the current state is s and action taken is a.

To reward behavior that we need to promote, we need to analyze both the present state of the agent, as well as the action undertaken. To analyze whether or not the state possesses desirable characteristics, we look at each feature of the state individually and observe the bin values that these features possess. It is also important to remember that for each feature, the values are increasing, for the increasing order of the bin. That is, for each feature's bins, bin 1 will have smaller numerical values than bin 2; bin 2 will have smaller numerical values than bin 3, and so on and so forth. Therefore, we positively reward high values of PRoPHET Probability (either bin 3 or bin 4) which is the first feature describing the state. This is done because a high value of probability is a strong positive indicator of eventual delivery of the message to the destination. Moreover, we positively reward low Buffer Occupancy (either bin 1 or bin 2) which is the second feature describing the MDP. This is done because more vacant buffers can store more messages for delivery and generally mean the network is congestion-free and overhead is low. We also positively reward high values (either bin 2 or bin 3) of successful message deliveries, which is the third feature of the MDP. The fourth feature describing the MDP, Distance To Destination also receives higher positive rewards when the physical distance between the relay node and the destination is small (either bin 1 or bin 2). For Message Live Time, the final feature describing the MDP, lower values (either bin 1 or bin 2) are given higher reward. This is done because a lower value of Message Live Time ensures that messages are being delivered in a timely manner, in turn, ensuring that network resource utilization and overhead are kept low.

Therefore, now that we know which behavior is desirable for the state, we formulate a mechanism for modeling it in the reward function. For each feature i, we assign an intermediate variable R_i (where $1 \leq i \leq 5$ for each feature's index number) with a binary value of either 1 or 0, depending on whether that feature is showing desirable characteristics for that state or not, respectively. This can be delineated as a set of conditional statements as follows:

- IF PRoPHET Probability (Feature 1) is HIGH (bin 3 or bin 4) $\Rightarrow R_1 = 1$ ELSE $R_1 = 0$
- IF Buffer Occupancy (Feature 2) is LOW (bin 1 or bin 2) $\Rightarrow R_2 = 1$ ELSE $R_2 = 0$
- If Successful Deliveries (Feature 3) is HIGH (bin 2 or bin 3) $\Rightarrow R_3 = 1$ ELSE $R_3 = 0$
- If Distance To Destination (Feature 4) is LOW (bin 1 or bin 2) $\Rightarrow R_4 = 1$ ELSE $R_4 = 0$
- If Message Live Time (Feature 5) is LOW (bin 1 or bin 2) $\Rightarrow R_5 = 1$ ELSE $R_5 = 0$

Now that we have the intermediate reward variables (R_i) for each feature i, we can begin to design the reward function. Since some features describing the states can have more importance than others, we design the reward function as a weighted sum using the intermediate variables. The weight values chosen for each feature are higher or lower depending on whether the feature is more important to the routing process and the optimization problem than the others. Thus, we assign weights to intermediate rewards of features depending on their relevance in the routing scenario as follows: R_1 (PRoPHET Probability) and R_4 (Distance To Destination) are weighted with a value of 2000 while all the other features are weighted with 1000 each. Here, it is important to understand that the actual numerical values of the weights are not as important, but the relative difference in values for these weights is noteworthy. That is, the fact that the PRoPHET Probability and Distance To Destination features are given more preference than the other features is more important than the values chosen for the weights themselves. Here, PRoPHET Probability is a strong measure of potential message delivery which is comparatively more reliable than the other features. This is because it is based on the frequency of encounters, and a transitivity property which ensures that node history and behavior are used to predict future delivery probability. Similarly, Distance to Destination is considered as a feature which is a strong indicator of eventual delivery. This is because proximity to the message destination increases the possibility of message delivery manifold. The other features are also important to the routing process, but are not sure-shot indicators of eventual message delivery. For example, the Successful Deliveries feature indicates that the node pair has a past of a high number of successful deliveries, however, there is no real guarantee that the message will be delivered for this encounter as well. Similarly Buffer Occupancy is a measure of occupied space in the node buffer, but even having practically empty buffers cannot guarantee eventual delivery as that is dependent on other external factors to a large extent. In another case, the buffer might be almost full, but certain extrinsic factors might ensure successful delivery. Therefore, the weights chosen are: $w_1 = 2000$, $w_2 = 1000$, $w_3 = 1000$, $w_4 = 2000$, and $w_5 = 1000$.

Thus, now we can write the reward that is obtained from the state, R_s as:

$$R_s = \sum_{i=1}^{n} R_i.w_i \tag{2}$$

where $n = 5$, as we have five features.

Next, we need to formulate how to assign rewards for the action taken by the agent. To do this, we need some measure to indicate whether the action taken was favorable or not. We need to take actions that increase message delivery probability, and decrease network resource utilization and overhead. Therefore, we can formulate a median value of the maximum and minimum possible rewards that can be obtained from the state, and if the choice of action displays desirable behavior upon comparison of it's present state reward and this median reward value, we can deem that action as favorable. Thus, we define the median state-reward mathematically, as follows:

$$\overline{R_s} = \frac{max(R_s) + min(R_s)}{2} \tag{3}$$

To determine the reward to be awarded according to the action taken, it is important to understand that the median state-reward $\overline{R_s}$ is indicative of the median reward that the agent can obtain in the state, whereas R_s is the actual reward obtained for that state. Here, we adjudge the desirability of the action undertaken using both these metrics. In simple terms, if the action chosen by the agent is 1 (which means that it has chosen not to forward the message), and for this scenario if $R_s \leq \overline{R_s}$, then this action is desirable. Since $R_s \leq \overline{R_s}$, the state-reward is very low and the features describing the state make it such that forwarding the message to the relay in the encounter would probably not have led to successful message delivery eventually. The 1 action for this scenario should thus be rewarded positively. However, in another case, when the action chosen is 1 (not choosing to forward the message) and $R_s \geq \overline{R_s}$, then the agent made an error as here the MDP state is favorable for routing and there is a high chance it would have led to successful delivery. Thus, the 1 action in this scenario should be rewarded negatively. In a similar vein, if the action chosen is 0 (choosing to forward the message) but $R_s \leq \overline{R_s}$, this means that the agent did not take the correct action as here chances of successful eventual delivery are very low. The action 0 in this scenario should also be awarded negative reward. However, in the case that the action undertaken is 0 and $R_s \geq \overline{R_s}$, the agent has displayed desirable behavior and it should be awarded with positive reward. This is because for this scenario there is a high chance of successful delivery and the agent has recognized this from the state feature set by taking action 0.

Since this value is unweighted, we label it as r_a and the final weighted action-reward as R_a. Moreover, when the action undertaken is as desired, we assign $r_a = 1$ and in case it is undesired and erroneous, we assign $r_a = -1$. The unweighted action-reward can thus be calculated using the following conditional statements:

- IF $Action = 1$:
  - IF $R_s \leq \overline{R_s} \Rightarrow r_a = 1$
  - IF $R_s \geq \overline{R_s} \Rightarrow r_a = -1$
- IF $Action = 0$:
  - IF $R_s \geq \overline{R_s} \Rightarrow r_a = 1$
  - IF $R_s \leq \overline{R_s} \Rightarrow r_a = -1$

Next, we also need to assign a weight to the action (unweighted) reward value that we obtain. We choose this weight value (w_a) in a rational manner by setting it to be the average of all the other state feature weights:

$$w_a = \frac{\sum_{i=1}^{n} w_i}{n} \tag{4}$$

Thus, we can now write the reward obtained from choosing the action R_a as:

$$R_a = r_a.w_a \tag{5}$$

Since we now have both the state-reward and the action reward, our reward function formulation is complete, and we can write the overall reward obtained by the agent, R, as:

$$R = R_s + R_a \tag{6}$$

3.1.4 State transition probabilities (P_t)

This function gives a measure of the likelihood of transition from one state to another. It can be mathematically represented as:

$$P_t(s, s') = \mathbb{P}(s_{t+1} = s' | s_t = s, a_t = a) \tag{7}$$

here P_t is the probability that action a in state s at time t will lead to state s' at time $t + 1$.

For assigning the state transition probabilities, we first analyze our basic requirements for the routing problem. Our main objectives are to maximize message delivery probability and also minimize the network overhead. Therefore, it is important to understand that for states which are not suitable for eventual message delivery, there should be high values of transition probabilities to other states. Similarly, states which possess characteristics ideal for message delivery, should also possess low values of transition probabilities to other states as the original states are ideal for routing themselves. Thus, if a device node is in a state which is ideal for routing, it should seek to remain there instead of transitioning to other states. Moreover, if a device node is in a state which will not allow optimal routing, it should have high transition probabilities to transition to other states which are more capable of effective message delivery.

To assign the transition probabilities in this manner, we use the state-reward metric defined and evaluated in the previous subsection (Eq. 2). Using the state-reward values we first compute the *inverse* of the state-reward (R_s') as $R_s' = max(R_s) - R_s$. Using this value for normalization makes it mathematically easier to assign high values of transition probabilities to states whose state-rewards are generally lower, for reasons we discussed in the previous paragraph. Moreover, after obtaining the *inverse* reward values, we simply normalize the R_s' values to between 0 and 1 while ensuring stochastic validity, which is a requirement for the transition probabilities in an MDP. This means that we have to ensure that for each row in the created transition probability matrix, the sum of the probabilities should add up to 1. Therefore, after normalization and assignment of transition probability to a state in a row, we equitably distribute the remaining probability to all the other states in the row to maintain stochastic validity.

In this manner, we are able to assign high transition probabilities to states which have low state-reward, and low transition probabilities to states which have high values of state-reward, while maintaining stochastic validity for the transition matrix that will hold the transition probability values.

3.1.5 Discount factor (γ)

The discount factor γ is the final characteristic of the MDP five-tuple. The discount factor is constrained between 0 and 1, that is, $\gamma \in [0, 1]$ and is a measure of whether the RL system prefers long-term reward over short-term reward. The closer γ is to 1, the more preference is given to long-term reward over short-term reward. In our approach and subsequent experiments, we did not notice any changes in the obtained optimal policy upon solving the MDP, for variations in the value of γ. However, since we want the optimal policy to prefer long-term reward over short-term immediate gain, we set the value of $\gamma = 0.9$.

3.2 Finding the optimal policy using policy iteration

3.2.1 The optimal policy

A policy (π) for a MDP specifies which action to execute when the agent is in a given state. In other words policy defines the way in which the agent would behave in a given state. Therefore, a policy for a MDP lists out the action to be executed corresponding to each state that makes up the MDP. A policy can be understood to be a vector containing actions (in our case, 0 or 1) where the length of the vector is the number of states in the MDP. Thus, mathematically, a MDP policy can be represented as:

$$\pi(a|s) = \pi(s) = \mathbb{P}[A_t = a|S_t = s] \tag{8}$$

An *optimal policy* (π^*), is a policy which seeks to maximize the cumulative reward for each possible state that the agent can be in. An optimal policy can thus be written as a summation of rewards accrued over an infinite horizon:

$$\pi^* = \sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1}) \tag{9}$$

where γ is the discount factor, a_t is the action we take according to the policy, that is, a_t=$\pi(s_t)$ and R represents the reward received when transitioning from state s_t to s_{t+1}.

3.2.2 Solving the MDP using policy iteration

We find the optimal policy for our MDP using the Policy Iteration algorithm [14]. Policy Iteration is a Dynamic Programming (DP) based algorithm which utilizes two recursive computation steps for computing the optimal policy. These two steps are: finding the policy ($\pi(s)$) for a state s and calculating the *value* for the same state s under policy $\pi(s)$ using the Bellman Value function, $V(s)$:

$$\pi(s) = argmax_a\left\{\sum_{s_{t+1}} P_a(s_t, s_{t+1})(R_a(s_t, s_{t+1}) + \gamma V(s_{t+1})\right\} \tag{10}$$

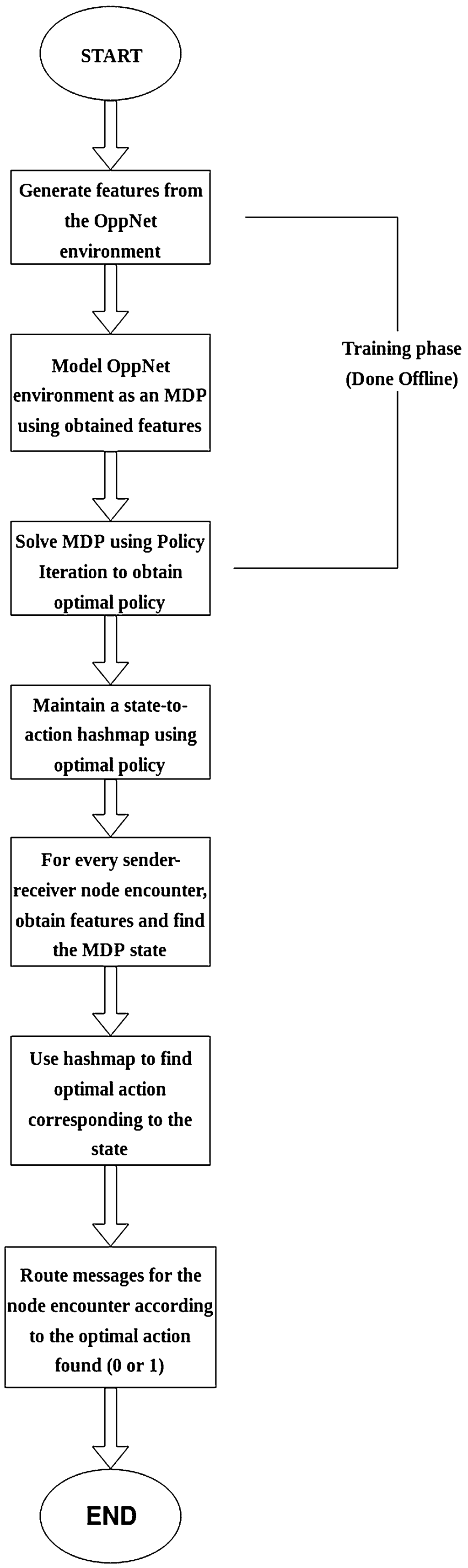
$$V(s) = \left\{\sum_{s_{t+1}} P_{\pi(s)}(s_t, s_{t+1})(R_{\pi(s)}(s_t, s_{t+1}) + \gamma V(s_{t+1}))\right\} \tag{11}$$

Generally, in Policy Iteration, the policy $\pi(s)$ (Eq. 9) is calculated first and then the Value function $V(s)$ (Eq. 10) is computed repeatedly till it converges. Then, the policy is re-computed and the same process (computing $V(s)$ till convergence) is repeated again, so on and so forth. These steps are repeated by the algorithm till the optimal policy π^* has been computed, that is, no further change takes place in the computed $\pi(s)$ for all states.

In this way, we obtain the optimal policy for our constructed MDP using Policy Iteration. This optimal policy as a consequence of maximizing the reward function, will in turn make the agent take decisions which maximize message delivery probabilities and route messages in an optimal fashion. The agent hence learns how to execute efficient routing in an OppIoT network in this manner. In the next section, we utilize this computed optimal policy for making intelligent routing decisions that increase the possibility of message delivery while minimizing network resource utilization.

3.3 Using the obtained optimal policy for routing in RLProph

The previous Sects. 3.1 and 3.2 detail how the MDP is constructed in our approach and how it is solved using Policy Iteration to obtain the optimal policy, respectively. Thus, these two sections actually cover the entire *training phase* for the RLProph framework, and lay the groundwork for improving the routing process using the obtained optimal policy.


START
Generate features from the OppNet environment
Model OppNet environment as an MDP using obtained features
Solve MDP using Policy Iteration to obtain optimal policy
Training phase (Done Offline)
Maintain a state-to-action hashmap using optimal policy
For every sender-receiver node encounter, obtain features and find the MDP state
Use hashmap to find optimal action corresponding to the state
Route messages for the node encounter according to the optimal action found (0 or 1)
END

◀**Fig. 2** Flowchart describing the routing framework of RLProph

This section details how the optimal policy is used to make intelligent routing decisions that improve overall delivery probability for the network and reduce unnecessary network overhead. To do this, we first maintain a hashmap of state-to-action values using the optimal actions listed in the optimal policy. As the optimal policy π^* is just a vector (of size equal to the number of states), containing actions 0 or 1 (decisions to either forward the message or not), we can transform it to a data structure such as the hashmap with states as the map *keys* and the actions as the key *values*. The use of this state-to-action hashmap will be enumerated in the routing algorithm subsequently.

In RLProph, we utilize the state-to-action hashmap for every sender-receiver node encounter to ascertain whether the device node should forward the message at this time or wait for some other relay and forward the message to them (action 0 or 1). However, before this can be done, we also need to find out the state of the MDP that the device node is in currently. Once the state is known, the optimal action can be obtained from the state-to-action hashmap. To obtain the state, we first extract the same five features used for constructing the MDP states, from this particular OppIoT node encounter. These are PRoPHET Probability, Buffer Occupancy, Successful Deliveries, Distance To Destination and Message Live Time. Once these continuous feature values are known, we discretize them using the equal-frequency binning algorithm shown in Algorithm 1, for four bin values for all features other than Successful Deliveries. For Successful Deliveries, three bins are used similarly to the MDP construction step. As can be seen, we are obtaining the MDP state for this real-time node encounter similar to the way we had obtained states for constructing the MDP in Sect. 3.1.1.

After using Algorithm 1 for binning the continuous feature values, we obtain the state corresponding to this particular sender-receiver node encounter. We then lookup the action to follow (0 or 1) for this state using the state-to-action hashmap. Moreover, now that we know which action to take, the node either forwards the message to the receiver, or does not forward the message to the receiver at this time, waiting for a better opportunity in the future. In this manner, this process is repeated for each node encounter in the OppIoT environment, and routing is accomplished. This complete routing algorithm can be seen in Algorithm 2.

The entire RLProph framework for routing is presented as a flowchart in Fig. 2.

Algorithm 2 RLProph Routing Protocol

```
1: for each encounter between sender-receiver node pair $(s, r)$ do
2:     extract five feature set $F$: (PRoPHET Probability, Buffer Occupancy, Successful Deliveries, Distance To
       Destination, Message Live Time) from $(s, r)$
3:     discretize $F$ using Equal-Frequency-Binning (Algorithm 1) and obtain MDP state $S$
4:     lookup optimal action $A \in \{0, 1\}$ from state-to-action hashmap $H(S, A)$ using $S$ as key
5:     if $A == 0$ then
6:         relay message
7:     else
8:         continue
```

4 Results analysis

We simulate RLProph using the Java based Opportunistic Network Environment (ONE) simulator [30]. The performance of RLPROPH is then compared with that of PRoPHET, HBPR, KNNR and Epidemic routing protocols on the basis of three key performance metrics, which are described as follows:

- *Message delivery probability* It indicates the number of messages successfully delivered to their destinations. A high delivery probability is desirable for an efficient routing protocol.
- *Number of packets dropped* It denotes the number of message packets dropped from the buffers of the nodes during routing. This happens when a node with a completely full buffer receives a relayed packet. An efficient routing protocol drops as few packets as possible.
- *Network overhead ratio* It signifies the average number of forwarded copies for each message and represents the overall overhead experienced by the network. A high overhead ratio is indicative of poorly managed storage resources. For efficient functioning of a routing protocol, a low overhead ratio is desirable since resources are limited.

Initially, we use the default settings mentioned in Table 2 to undertake the comparative analysis between the routing protocols. We then vary selected parameters that can affect the functioning of the routing protocol, and carry out comparative analysis on the obtained simulation results. These selected parameters are enlisted below:

- *Message time-to-live (TTL)* This can be described as the maximum amount of time a message can survive in the network after its generation. If a message is not successfully delivered to its destination within this period of time, it is considered as invalid and is dropped from all the nodes in the network. The TTL of the messages is varied as: 50, 100, 150, 200, 250, 300 (Unit:Minutes).
- *Simulation time* This is total duration of time for which a particular simulation is run. The simulation time is varied as: 10,800, 21,600, 32,400, 43,200 (Unit:Seconds).

Table 2 Default simulation settings

Simulation parameter	Value
Simulation time	43200 s
Common tnterface	Bluetooth (BT)
Tram tnterface	High-speed (HS)
HS transmit speed	10 MBps
BT transmit speed	1 MBps
HS transmit range	1000 m
BT transmit range	100 m
Pedestrian group(s)	2
Car group(s)	1
Tram group(s)	3
Pedestrian speed	4.5–6.5 m/s
Pedestrian wait time	0–120 s
Car speed	2.7–13.9 m/s
Tram speed	7–10 m/s
Tram wait time	10–30 s
Tram movement model	Map route movement
Message generation Interval	15–25 s
Message size	500kB–1 MB
Host groups	6
Movement model	Shortest path map based movement
Host buffer size	10 MB
Message TTL	300 min
Host initial energy	18 kJ
Host scan energy	0.4 J
Host transmit energy	0.5 J
Host scan response energy	0.4 J
Host base energy	0.04 J

- *Message generation interva* The message generation interval is varied as follows: between 5–15, 15–25, 25–35, 35–45, 45–55 s (Unit:Seconds). A new message is generated every *t* seconds, where *t* is a randomly chosen time period lying between the lower and upper bounds of the given interval. That is, for 5–15 s for example, a new message will be randomly generated after a period of time which lies between 5 and 15 seconds.

4.1 Results on default settings

First, we compare the aforementioned protocols: Epidemic, PRoPHET, KNNR, and HBPR with RLProph on default settings of the ONE simulator. These settings are listed in Table 2. The results obtained are described in Table 3. It can be seen that RLProph gives excellent results and outperforms all the other protocols in terms of message delivery probability and network overhead ratio. The message delivery probability of RLProph is 47.78% which is much higher than all the other protocols. The second highest delivery probability is achieved by PRoPHET which is 41.51%. The reason for the highest delivery probability could be traced back to the way the reward function has been designed, and how it promotes higher message delivery states for nodes. Thus, RLProph surpasses other protocols in delivery probability. This can also be observed in Fig. 3(b). The confidence level of the computed delivery probability is 99%.

RLPRoph also achieves the lowest network overhead ratio amongst all the other protocols. RLProph obtains an overhead ratio of 54.2144 followed by KNNR which obtains an overhead ratio of 68.1621. This is due to the fact that the feature parameters used for defining the MDP states, such as Message Live Time and Buffer Occupancy are considered by the reward function while assigning rewards to the MDP states. A message that generally exists for a very long time in the network decreases node energy without leading to delivery and exhausts network resources. This in turn increases the network overhead. Moreover, overpopulated buffers also indicate that the network is being over-utilized. Since the reward function is inclined to awarding more reward for states where the Message Live Time is not high, and where Buffer Occupancy is generally low, the network overhead (ratio) should be optimally low as a result. The observed results prove this inference to be correct. This is also corroborated by the obtained results, as can be seen in Fig. 4(a). The confidence level of the observed value of overhead ratio is 99%.

In the case of packets dropped, RLProph ends up at second position with 63777 packets dropped. KNNR has the least number of packets dropped, which is 51133. This value of the number of packets dropped also includes the number of packets that were generated by the routing protocol and then dropped. Since KNNR generates a meagre number of packets according to its protocol (65615), the number of messages dropped is also very less. RLProph on the other hand generates a larger number of packets (81375). Since the number of packets generated are more, the number of packets dropped are slightly more (than KNNR). The results can be observed in Fig. 3(a). However this figure is not truly indicative of the comparison of the performance of RLProph and KNNR on this parameter. This can be confirmed by calculating the ratio of message dropped to message generated. For RLProph this value comes out to be around 0.783, while for KNNR this value is 0.779. Hence had KNNR generated equivalent number of messages as RLProph, the difference in the corresponding values of message drops would have been very low. Moreover since RLProph is able to exhibit much higher value of delivery probability, it becomes the more reliable and efficient routing protocol. The number of packets dropped have been observed at a confidence level of 99%.

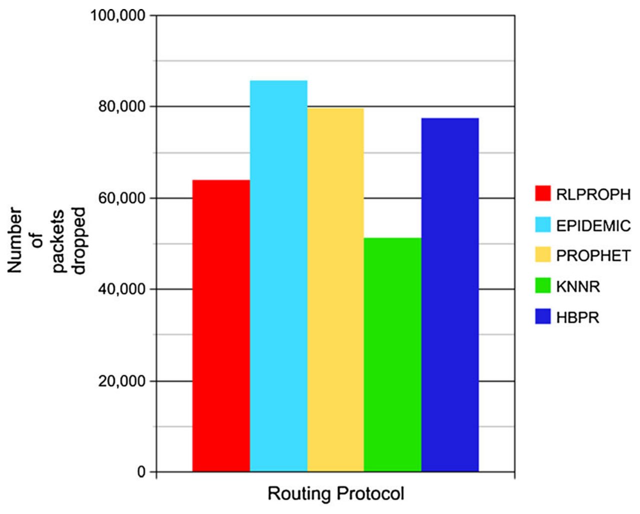
In the forthcoming sub sections we further delve into the comparative analysis of the simulation results obtained by changing certain selected parameters, mentioned previously in Sect. 4, for all of the routing protocols under consideration.

4.2 Results on varying simulation time

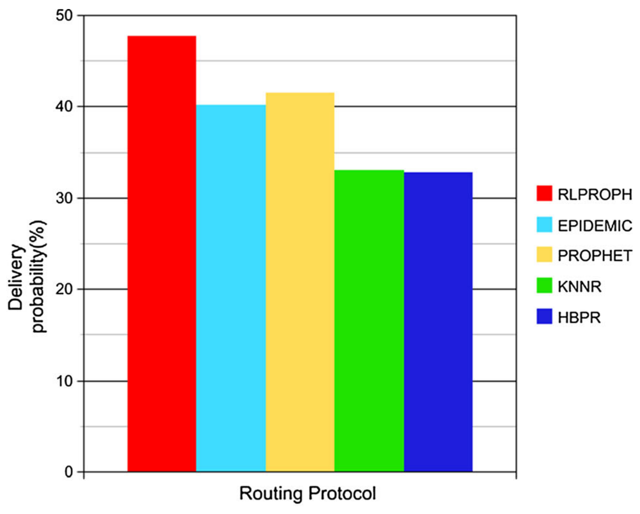
The comparative analysis in this section is done by changing simulation time for all the protocols. This has been done to test the performance of RLProph both in the short run as well as long run. Figure 4(b) depicts the results obtained for the number of packets dropped while varying

Table 3 Results for performance metrics on the default settings

Routing protocol	Message delivery probability (%)	Packets dropped	Network overhead ratio
RLProph	**47.68**	**63,777**	**59.2144**
KNNR	33.05	**51,133**	68.1621
Epidemic	40.16	85,712	95.0493
PRoPHET	41.51	79,594	85.2907
HBPR	32.82	77,323	104.6626

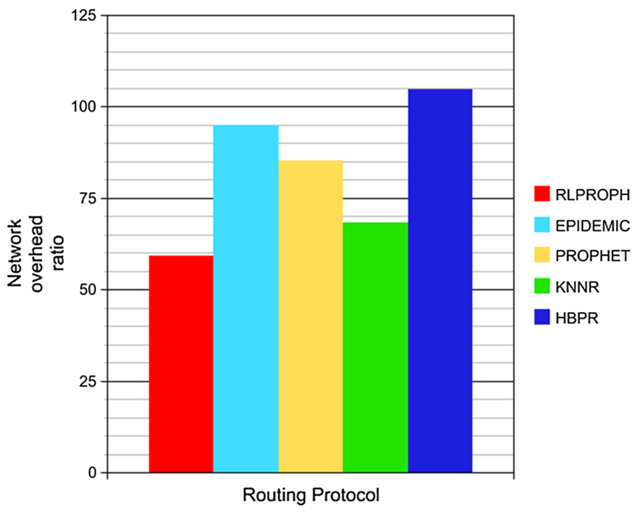


(a)

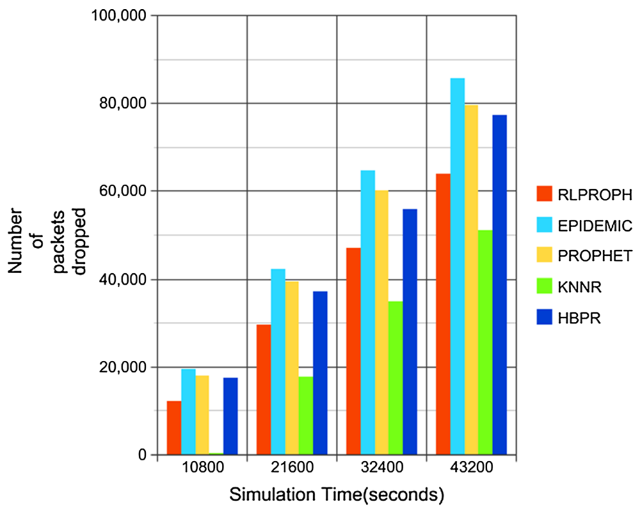


(b)

Fig. 3 **a** Packets dropped. **b** Delivery probability



(a)



(b)

Fig. 4 **a** Overhead ratio. **b** Packets dropped versus simulation time

simulation time. The initial value for 10800 seconds can be discarded for KNNR because at that time KNNR has not started transmitting messages and is still in its training phase. It is clear that RLProph outperforms every protocol except for KNNR for this metric due to the fact that KNNR generates comparatively fewer messages (500, 23,555, 44,780, 65,615) compared to RLProph (16,314, 38,356, 60,264, 81,375). As simulation time is varied from 10,800 to 43,200 s, RLProph drops 12,093, 29,537, 46,949, 63,777 number of packets. KNNR marginally beats these results by dropping N/A, 17713, 34739, 51133 number of packets. As mentioned in Sect. 4.1, these values are not the true indicators of the comparison between KNNR and RLProph on the performance metric of number of packets dropped. The ratio of message dropped to message generated for KNNR (N/A, 0.75, .78, 0.78) is almost equal to that of RLProph (0.74, 0.77, 0.78, 0.78). Hence, had KNNR generated more messages, we would have seen a higher packet drop count for it. Epidemic routing protocol drops between (19,337–85,712) packets during its simulation and PRoPHET drops between (18,030–79,594) packets.

Figure 5(a) shows the results obtained for delivery probability while varying simulation time. We can ascertain from this graph that RLProph shows optimal values of delivery probability in almost all cases. At 10800 seconds KNNR performs slightly better compared to RLProph. This is because RLProph makes routing decisions that improve the delivery probability in the long-run compared to KNNR. KNNR can be seen to make certain choices at the start which initially lead to some messages delivered but

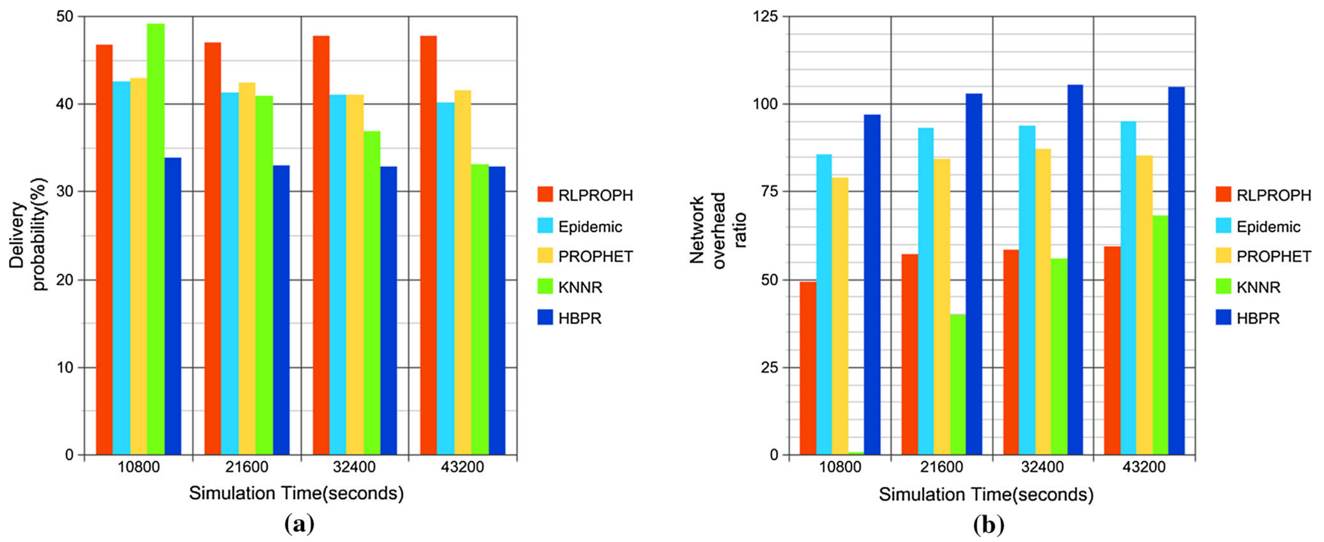


Fig. 5 **a** Delivery probability versus simulation time. **b** Overhead ratio versus simulation time

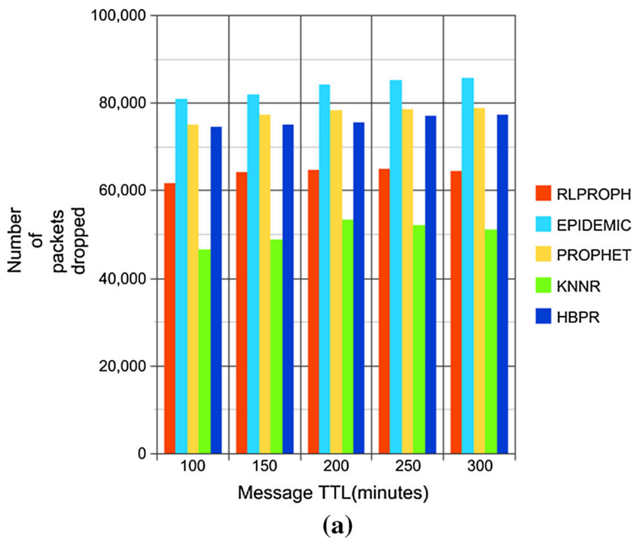
eventually do not yield good results. Eventually, RLProph has the highest values of delivery probability compared to all the other routing protocols. The values of delivery probability for RLProph for (10,800–43,200) seconds are: (46.74%, 46.96%, 47.75%, 47.68%). For KNNR the values are: (49.09%, 40.89%, 36.83%, 33.05%), hence proving that RLProph works better in the long-term. The results obtained for PRoPHET are: (42.43%, 41.03%, 41.51%, 46.74%) whereas these values for Epidemic routing are as follows: (42.57%, 41.25%, 41.09%, 40.16%).

Figure 5(b) denotes the results for the overall network overhead ratio varied with simulation time. Here we can observe that KNNR gives a better overhead ratio for the first three-fourths of simulation, from (10,800–32,400) seconds but RLProph eventually excels and obtains optimal results at 43,200 seconds. Here too, the first value (at 10,800 s) for KNNR can be discarded since it is not transmitting any messages in its training phase. The values of network ratio for KNNR vary as (N/A, 39.6408, 55.8681, 68.1621). Whereas for RLPRoph these values are: (49.1008, 57.1622, 58.4108, 59.2144). At 43,200 s of simulation time, or 12 h, RLProph gives a significantly better overhead ratio compared to the other protocols. The reason for such performance from RLProph is that it seeks to maximize long-term reward by lowering network overhead. The values of network overhead ratio for PRoPHET protocol lie between (78.9536–85.2907), those of HBPR fall in the range of (96.7807–104.6626), and for Epidemic the range is (85.4043–95.0493).

4.3 Results on varying message time-to-live (TTL)

This section delineates the comparative analysis of the routing protocols of interest on the basis of simulations with varying message time to live. This has been done to check how the performance of RLProph is affected by the change in tolerance to delay of the network. This analysis can be useful in understanding if RLProph can be used in scenarios which require urgent message deliveries. Figure 6(a) denotes the number of packets dropped while varying message TTL. Results for RLProph are: (61,629, 64,118, 64,720, 64,922, 64,440). The reason for obtaining second-best results (to KNNR) for this metric are the same as mentioned in the previous subsections. Since KNNR generates only a very small number of messages (60,069, 62,934, 68,042, 66,704, 65,615) compared to RLProph (77,929, 81,743, 82,366, 82,684, 82,058), the overall dropped messages for it are low. The values of message drops for KNNR are: (46,563, 48,743,53,185, 52,002, 51,133). The value of message drops for the rest of the routing protocols vary as follows: Epidemic (80,804–85,712), HBPR (74,393–77,323), PRoPHET (75,119–78,859).

Figure 6(b) shows the results for delivery probability while varying message TTL. It can be seen that RLProph outperforms all of the other protocols with its state-of-the-art results: (49.98%, 4750%, 47.68%, 48.00%, 47.55%). However, at 100 seconds, PRoPHET performs slightly better than RLProph. This is only marginally better than RLProph, and it can be seen that PRoPHET is unable to sustain this optimal performance throughout the simulation. The value of deliver probability for PRoPHET are obtained as: (51.10%, 45.52%, 42.77%, 43.27%, 42.28%).



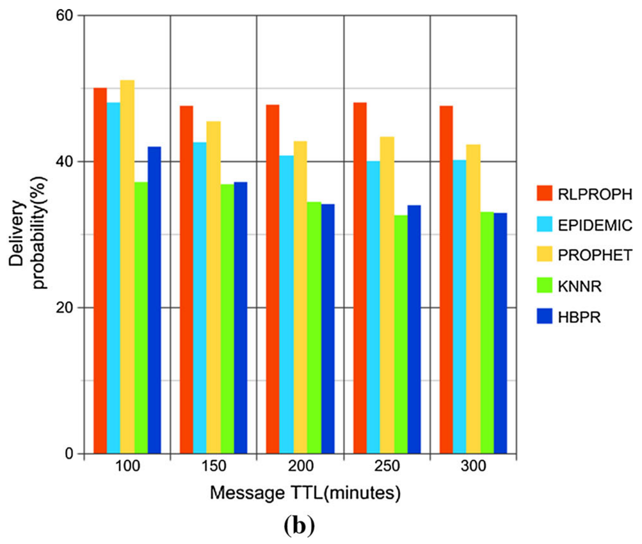
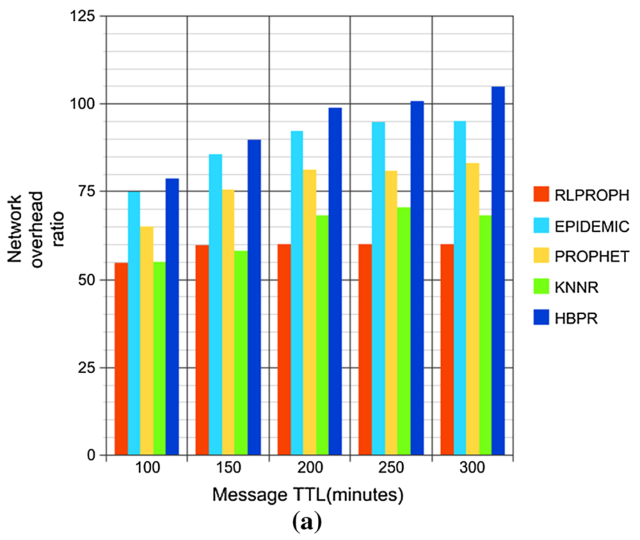


Fig. 6 **a** Packets dropped versus message TTL. **b** Delivery probability versus Message TTL



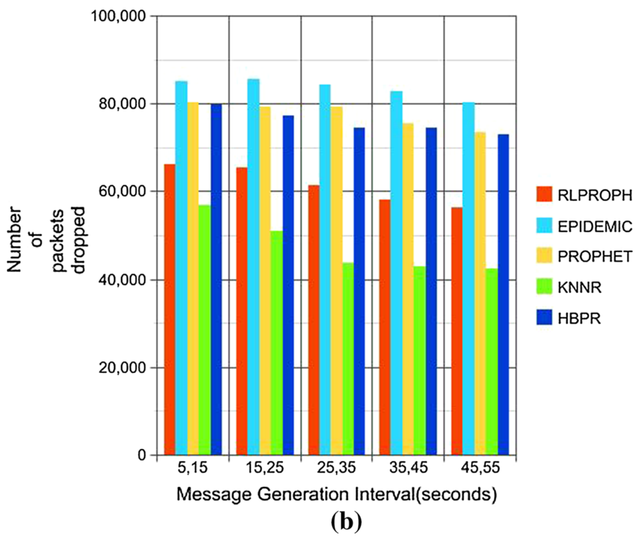


Fig. 7 **a** Overhead ratio versus message TTL. **b** Packets dropped versus message generation interval

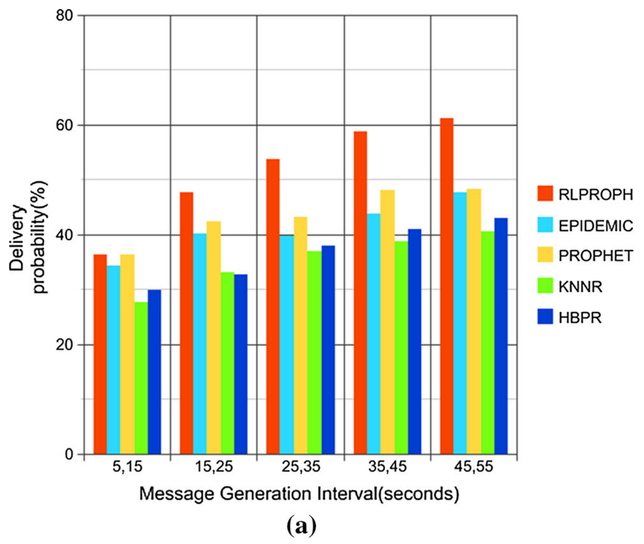
This value varies for the rest of the protocols in the following way: HBPR (41.92–32.82%), Epidemic (48.00–40.16%), KNNR (37.15–33.05%).

Figure 7(a) shows the results for the network overhead ratio while varying message TTL. Clearly, RLProph surpasses all routing protocols with value of network overhead ratio varying as: (54.5054, 59.7649, 60.1010, 59.8846, 59.9934). KNNR is the second-best performing protocol with results as follows: (54.8739, 58.1956, 68.1085, 70.3688, 68.1621). The performance of the rest of the protocols does not fare well in comparison to RLProph: Epidemic (74.8096–95.0493), HBPR (78.6842–104.6626), PRoPHET (65.1692–82.9180).

4.4 Results on varying message generation interval

This section describes the performance comparison of the routing protocols by varying message generation interval time in their simulations. This will be useful to analyse how RLProph performs in handling varying volumes of message traffic.

Figure 7(b) denotes the number of packets dropped while varying message generation interval. Results for RLProph are: (66,182, 65,411, 61,401, 58,053, 56,402). KNNR again “seemingly” performs better for this metric with the lowest values: (56,827, 51,133, 43,723, 43,051, 42,541), because it generates very few messages (70,045,



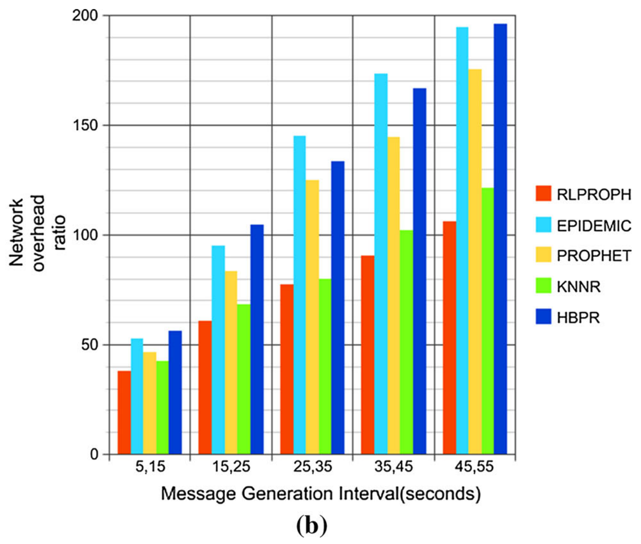


Fig. 8 **a** Delivery probability versus message generation interval. **b** Overhead ratio versus message generation interval

65,615, 56,987, 56,645, 55,698) compared to RLProph (82,437, 83,026, 78,945, 75,136, 73,110). The value of packets dropped for the other routing protocols vary as follows: Epidemic (85,024–80,357), HBPR (79,728–72,933), PRoPHET (80,207–73,466).

Figure 8(a) displays the results for delivery probability while varying the message generation interval. Here again, RLProph outperforms the other routing protocols, with the following results: (36.37%, 47.77%, 53.82%, 58.85%, 61.31%). PRoPHET shows the second best results with delivery probability values as follows: (36.28%, 42.41%, 43.18%, 48.03%, 48.34%). The rest of the protocols do not come close to RLProph in comparison with delivery probability values varying as follows: HBPR (29.92–42.94%), Epidemic (34.39–47.65%), KNNR (27.77–40.64%).

Figure 8(b) represents the network overhead ratio while the message generation interval is varied. Clearly, RLProph shows the most optimal results compared to all other routing protocols with the following values: (38.0145, 60.6371, 77.4385, 90.6495, 106.2528). Here the other protocols do not perform well, and obtain the following results: KNNR (42.2956–121.0254), Epidemic (52.3167–194.6313), HBPR (56.2279–196.0080), PRoPHET (46.6343–175.4228).

Thus it can be clearly seen that RLProph exhibits optimal performance in terms of the performance metrics, and achieves state-of-the-art results for message delivery probability, number of packets dropped and network overhead ratio: both for the default settings, and while different network parameters are varied.

4.5 Results from simulation on real mobility traces with different discretization setting

This final section includes the results from simulations run on ONE using the CRAWDAD [34] mobility traces dataset. This data set contains bluetooth traces of mobile devices carried by users for a number of days—in Intel Research Cambridge Corporate Laboratory, Computer Lab at University of Cambridge, Conference IEEE Infocom in Grand Hyatt Miami, and locations around the city of Cambridge, UK. This is done to see how RLProph will perform in a real world setting. As described in Sect. 3.1.1 the method of discretization of feature space to describe the states in the designed MDP is flexible and upto the designer of the network. To prove that RLProph still showcases outstanding performance when this method is changed, we ran simulations on ONE using the following feature discretization settings:

1. Feature 1 (PRoPHET Probability) $\rightarrow$ 5 bins
2. Feature 2 (Buffer Occupancy) $\rightarrow$ 5 bins
3. Feature 3 (Successful Deliveries) $\rightarrow$ 3 bins
4. Feature 4 (Distance To Destination) $\rightarrow$ 5 bins
5. Feature 5 (Message Live Time) $\rightarrow$ 5 bins

The conditional statements used to design the reward function in Sect. 3.1.3 are changed to the following to incorporate for the increase in number of bins for most of the features:

- IF PRoPHET Probability (Feature 1) is HIGH (bin 4 or bin 5) $\Rightarrow R_1 = 1$ ELSE $R_1 = 0$

- IF Buffer Occupancy (Feature 2) is LOW (bin 1 or bin 2) $\Rightarrow R_2 = 1$ ELSE $R_2 = 0$
- If Successful Deliveries (Feature 3) is HIGH (bin 2 or bin 3) $\Rightarrow R_3 = 1$ ELSE $R_3 = 0$
- If Distance To Destination (Feature 4) is LOW (bin 1 or bin 2) $\Rightarrow R_4 = 1$ ELSE $R_4 = 0$
- If Message Live Time (Feature 5) is LOW (bin 1 or bin 2) $\Rightarrow R_5 = 1$ ELSE $R_5 = 0$

We holistically compare the performance of RLProph against a wide variety of protocols. For this experiment RLProph was compared against both ML based context-aware protocol: KNNR and context-oblivious protocols: Epidemic and Spray and Wait.

Figures 9, 10 and 11 exhibit the results of the simulation run on the aforementioned settings. As is evident from Figure 9. RLProph has the least number of message drops when compared to Epidemic, Spray and Wait and KNNR routing protocols. Figure 10 is proof of the optimal resource utilization of RLProph as compared to other protocols. This shows that the RL based trained nodes have learned how to judiciously use network resources. Figure 10 showcases the comparison of the delivery probability values of the four protocols under comparison. Here too RLProph surpasses all the protocols. This proves that the RL agents have efficiently learned how to maximize delivery probability.

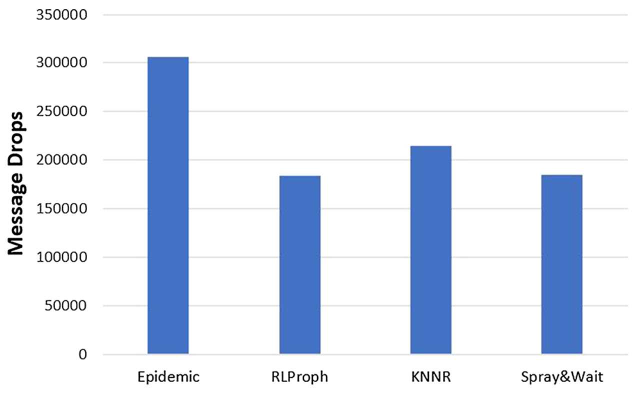


Fig. 9 Message drops comparison

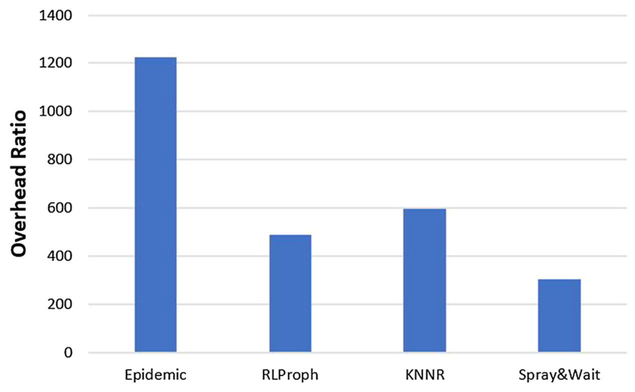


Fig. 10 Overhead ratio comparison

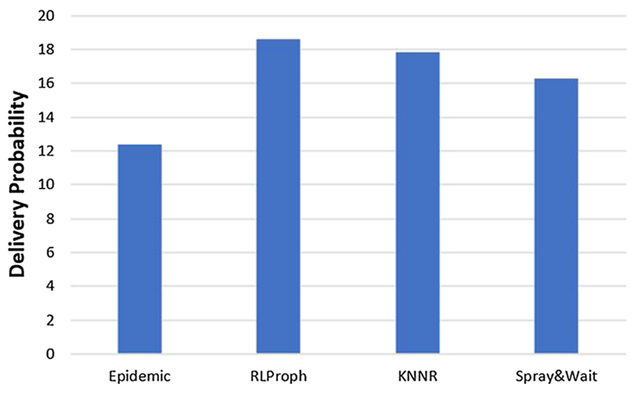


Fig. 11 Delivery probability comparison

5 Conclusion and future work

In this paper, we present a novel approach for routing in OppIoT networks, named RLProph, which uses a planning based RL approach for routing. Unlike traditional routing protocols, we model the OppIoT environment as an MDP and obtain an optimal policy by solving the MDP using the Dynamic Programming based iterative algorithm: Policy Iteration. The optimal policy is used by the routing process to make intelligent and efficacious routing decisions that improve the message delivery probability, while reducing network overhead. We compare RLProph with other routing protocols (KNNR, PRoPHET, HBPR, Epidemic and Spray Wait) via simulations on the ONE simulator and find that RLProph outperforms them on a wide variety of performance metrics.

As future work, we would like to work on developing innovative RL based techniques for securing OppIoT networks, and/or improving existing techniques for preventing blackhole attacks [35, 36]. We would also like to work on energy incentivization to curb selfish node behavior [37] by augmenting them with RL. We would also like to undertake work in further improving OppIoT routing by employing other RL approaches such as Policy Gradients [38] and Double Q-Learning [39].

Acknowledgements This work is partially supported by FCT/MCTES through national funds and when applicable co-funded EU funds under the Project UIDB/EEA/50008/2020; and by Brazilian National Council for Research and Development (CNPq) via Grant No. 309335/2017-5.

References

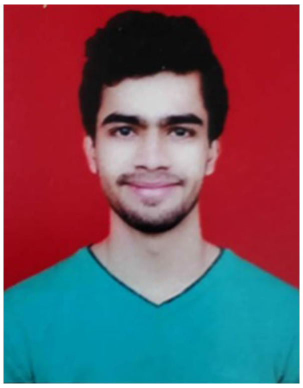
1. Juang, P., Oki, H., Wang, Y., Martonosi, M., Peh, L. S., & Rubenstein, D. (2002). Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet. *ACM SIGARCH Computer Architecture News*, *30*(5), 96–107.
2. Pentland, A., Fletcher, R., & Hasson, A. (2004). Daknet: Rethinking connectivity in developing nations. *Computer*, *37*(1), 78–83.
3. Doria, A., Uden, M., & Pandey, D. P. (2002). Providing connectivity to the saami nomadic community. *International Conference on open collaborative design for sustainable innovation*. 01/12/2002-02/12/2002.
4. Guo, B., Zhang, D., Wang, Z., Yu, Z., & Zhou, X. (2013). Opportunistic IoT: Exploring the harmonious interaction between human and the internet of things. *Journal of Network and Computer Applications*, *36*(6), 1531–1539.
5. Vahdat, A., & Becker, D. (2000). Epidemic routing for partially connected ad hoc networks. Technical Report CS-200006, Duke University. Available from: https://cloudcoder.cs.duke.edu/techreports/2000/2000-06.ps.
6. Lindgren, A., Doria, A., & Schelén, O. (2003). Probabilistic routing in intermittently connected networks. *SIGMOBILE Mobile Computer Communication and Review.*, *7*(3), 19–20. https://doi.org/10.1145/961268.961272.
7. Dhurandher, S. K., Sharma, D. K., Woungang, I., & Bhati, S. (2013). HBPR: History based prediction for routing in infrastructure-less opportunistic networks. In *2013 IEEE 27th international Conference on advanced information networking and applications (AINA)* (pp. 931–936).
8. Dhurandher, S. K., Sharma, D. K., Woungang, I., & Saini, A. (2015). Efficient routing based on past information to predict the future location for message passing in infrastructure-less opportunistic networks. *The Journal of Supercomputing*, *71*(5), 1694–1711.
9. Sharma, D. K., Dhurandher, S. K., Woungang, I., Srivastava, R. K., Mohananey, A., & Rodrigues, J. J. (2016). A machine learning-based protocol for efficient routing in opportunistic networks. *IEEE Systems Journal*, *12*(3), 2207–2213.
10. Vashishth, V., Chhabra, A., & Sharma, D. K. (2019). GMMR: A Gaussian mixture model based unsupervised machine learning approach for optimal routing in opportunistic IoT networks. *Computer Communications*, *134*, 138–148.
11. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., & Wierstra, D., et al. (2013). Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.
12. Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., et al. (2017). Mastering the game of Go without human knowledge. *Nature*, *550*(7676), 354.
13. Bellman, R. (1957). A Markovian Decision Process. *Indiana University Mathematics Journal*, *6*, 679–684.
14. Telser, L. G. (1961). Dynamic Programming and Markov Processes. Ronald A. Howard. *Journal of Political Economy.*, *69*(3), 296–297. https://doi.org/10.1086/258477.
15. Huang, T. K., Lee, C. K., & Chen, L. J. (2010). Prophet+: An adaptive prophet-based routing protocol for opportunistic network. In *2010 24th IEEE international Conference on advanced information networking and applications (AINA)* (pp. 112–119). IEEE.
16. Nguyen, H. A., Giordano, S., & Puiatti, A. (2007). Probabilistic routing protocol for intermittently connected mobile ad hoc network (propicman). In *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks* (pp. 1–6). Available from: https://ieeexplore.ieee.org/abstract/document/4351696/.
17. Leguay, J., Friedman, T., & Conan, V. (2005). MobySpace: Mobility pattern space routing for DTNs. In *ACM SIGCOMM Conference*.
18. Burns, B., Brock, O., & Levine, B. N. (2005). MV routing and capacity building in disruption tolerant networks. In *Proceedings IEEE 24th annual joint Conference of the IEEE computer and communications societies* (vol. 1, pp. 398–408).
19. Boldrini, C., Conti, M., Jacopini, J., & Passarella, A. (2007). HiBOp: A history based routing protocol for opportunistic networks. In *2007 IEEE international Symposium on a world of wireless, mobile and multimedia networks* (pp. 1–12).
20. Dhurandher, S. K., Borah, S. J., Obaidat, M. S., Sharma, D. K., Gupta, S., & Baruah, B. (2015). Probability-based controlled flooding in opportunistic networks. In: *2015 12th International joint Conference one-Business and telecommunications (ICETE)* (vol. 6, pp. 3–8). IEEE.
21. Xiao, M., Wu, J., & Huang, L. (2014). Community-aware opportunistic routing in mobile social networks. *IEEE Transactions on Computers*, *63*(7), 1682–1695.
22. Sharma, D. K., Sharma, A., & Kumar, J., et al. (2017). KNNR: K-nearest neighbour classification based routing protocol for opportunistic networks. In: *2017 Tenth international Conference on contemporary computing (IC3)* (pp. 1–6). IEEE.
23. Derakhshanfard, N., Sabaei, M., & Rahmani, A. (2017). CPTR: Conditional probability tree based routing in opportunistic networks. *Wireless Networks*. https://doi.org/10.1007/s11276-015-1136-4.
24. Zhang, J., Huang, H., et al. (2019). Destination-aware metric based social routing for mobile opportunistic networks. *Wireless Networks*. https://doi.org/10.1007/s11276-018-01907-2.
25. Jang, K., Lee, J., et al. (2016). An adaptive routing algorithm considering position and social similarities in an opportunistic network. *Wireless Networks*. https://doi.org/10.1007/s11276-015-1048-3.
26. Malekyan, S., Bag-Mohammadi, M., et al. (2019). On selection of forwarding nodes for long opportunistic routes. *Wireless Networks*. https://doi.org/10.1007/s11276-017-1636-5.
27. Wu, J., Chen, Z., & Zhao, M. (2019). Information cache management and data transmission algorithm in opportunistic social networks. *Wireless Networks*. https://doi.org/10.1007/s11276-018-1691-6.
28. Spyropoulos, T., Psounis, K., & Raghavendra, C. S. (2005). Spray and wait: An efficient routing scheme for intermittently connected mobile networks. In *Proceedings of the 2005 ACM SIGCOMM Workshop on Delay-tolerant networking* (pp. 252–259). ACM.
29. Spyropoulos, T., Psounis, K., & Raghavendra, C. S. (2007). Spray and focus: Efficient mobility-assisted routing for heterogeneous and correlated mobility. In *Null*, (pp. 79–85). IEEE.
30. Keränen, A., Ott, J., & Kärkkäinen, T. (2009). The ONE simulator for DTN protocol evaluation. In *Proceedings of the 2nd international conference on simulation tools and techniques. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering)* (pp. 55).

31. Fayyad, U., & Irani, K. (1993). Multi-interval discretization of continuous-valued attributes for classification learning. Available from: https://trs.jpl.nasa.gov/handle/2014/35171.
32. Kurgan, L. A., & Cios, K. J. (2004). CAIM discretization algorithm. *IEEE Transactions on Knowledge and Data Engineering*, *16*(2), 145–153.
33. Tsai, C. J., Lee, C. I., & Yang, W. P. (2008). A discretization algorithm based on class-attribute contingency coefficient. *Information Sciences*, *178*(3), 714–731.
34. Scott, J., Gass, R., Crowcroft, J., Hui, P., Diot, C., & Chaintreau, A. (2009). CRAWDAD dataset cambridge/haggle (v. 2009-05-29); Downloaded from https://crawdad.org/cambridge/haggle/20090529.Accessed 6 Jan 2020.
35. Chhabra, A., Vashishth, V., & Sharma, D. K. (2017). A game theory based secure model against Black hole attacks in Opportunistic Networks. In *2017 51st Annual conference on information sciences and systems (CISS)*, (pp. 1–6). IEEE.
36. Chhabra, A., Vashishth, V., & Sharma, D. K. (2018). A fuzzy logic and game theory based adaptive approach for securing opportunistic networks against black hole attacks. *International Journal of Communication Systems.*, *31*(4), e3487.
37. Chhabra, A., Vashishth, V., & Sharma, D. K. (2017). SEIR: A Stackelberg game based approach for energy-aware and incentivized routing in selfish Opportunistic Networks. In *2017 51st Annual Conference on information sciences and systems (CISS)*, (pp. 1–6). IEEE.
38. Sutton, R. S., McAllester, D. A., Singh, S. P., & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems* (pp. 1057–1063).
39. Van Hasselt, H., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Thirtieth AAAI Conference on Artificial Intelligence*. Available from: https://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/viewPaper/12389.

Deepak Kumar Sharma received the B.Tech. in Computer Science and Engineering, M.E. in Computer Technology and Applications and Ph.D. in Computer Engineering from University of Delhi, India. He is presently working as Assistant Professor in the Department of Information Technology, Netaji Subhas University of Technology (Formerly N.S.I.T.), Dwarka, New Delhi, India. His current research interests include opportunistic networks, wireless ad hoc networks sensor networks, Fog Computing and IoT. He has over 15 years of experience in Academics and guiding four Ph.D. students. He has published several papers in reputed Journals and conferences proceedings and has also authored chapters in various books.

Joel J. P. C. Rodrigues [S'01, M'06, SM'06, F'20] is a professor at the Federal University of Piauí, Brazil; senior researcher at the Instituto de Telecomunicações, Portugal; and collaborator of the Post-Graduation Program on Teleinformatics Engineering at the Federal University of Ceará (UFC), Brazil. Prof. Rodrigues is the leader of the Next Generation Networks and Applications (NetGNA) research group (CNPq), an IEEE Distinguished Lecturer, Member Representative of the IEEE Communications Society on the IEEE Biometrics Council, and the President of the scientific council at ParkUrbis - Covilhã Science and Technology Park. He was Director for Conference Development - IEEE ComSoc Board of Governors [2018–2019], Technical Activities Committee Chair of the IEEE ComSoc Latin America Region Board [2018–2019], a Past-Chair of the IEEE ComSoc Technical Committee on eHealth, a Past-chair of the IEEE ComSoc Technical Committee on Communications Software, a Steering Committee member of the IEEE Life Sciences Technical Community and Publications co-Chair [2014–2017]. He is the editor-in-chief of the International Journal on E-Health and Medical Communications and editorial board member of several high-reputed journals. He has been general chair and TPC Chair of many international conferences, including IEEE ICC, IEEE GLOBECOM, IEEE HEALTHCOM, and IEEE LatinCom. He has authored or coauthored over 850 papers in refereed international journals and conferences, 3 books, 2 patents, and 1 ITU-T Recommendation. He had been awarded several Outstanding Leadership and Outstanding Service Awards by IEEE Communications Society and several best papers awards. Prof. Rodrigues is a member of the Internet Society, a senior member ACM, and Fellow of IEEE.

Vidushi Vashishth has a degree in Bachelors of Engineering in Information Technology, from Netaji Subhas Institute of Technology, University of Delhi, batch of 2018. Her research interests include mobile ad-hoc networks, Internet of things networks and cloud computing. She has authored several papers in these fields. She is currently working as a software developer in Adobe Systems Pvt Lmt.



Anirudh Khanna is an Electronics and Communication Engineering graduate from Netaji Subhas University of Technology, University of Delhi, India. He is currently working with the Data science team of a FinTech Startup. His research interests include disconnected wireless networking, sensor networks, Reinforcement learning, NLP and algorithmic computer science.

Anshuman Chhabra is a Ph.D. student in the Department of Computer Science at the University of California, Davis. His research interests include Internet-of-Things, Networked systems, and Blockchain. He is also interested in the application of stochastic game theory and online learning to these fields, especially in the design and development of self-configuring-networks and automated network security.